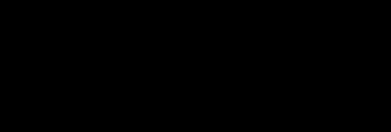


Drawing Text With ATSUI

Session 202

Drawing Text With ATSUI

Xavier Legros
Mac OS X Evangelist
Apple Worldwide Developer Relations

Overview

- Introduction to ATSUI
    - Features
    - Concepts
- New API For Jaguar
- Dos and Don’ts

What Is ATSUI?

- Mac OS X basic text drawing API
  - Only way to draw Unicode and the best way to get Quartz for Carbon apps

ATSUI Features

- Full Unicode 3.2 Layout Support
  - Truly Multilingual—one set of API to support all languages
  - Combining characters and complex scripts
  - Languages not covered by WorldScript I or II
  - Replaces WorldScript I on Mac OS X
- Automatic Font Substitution

More ATSUI Features

- Advanced typography
  - Kerning, optical alignment, variation fonts, ligatures, glyph alternatives, baseline adjust, . . .
  - Vertical text layout (CJK)
- Editing
  - Highlighting, hit testing, cursor movement
- Quartz Drawing

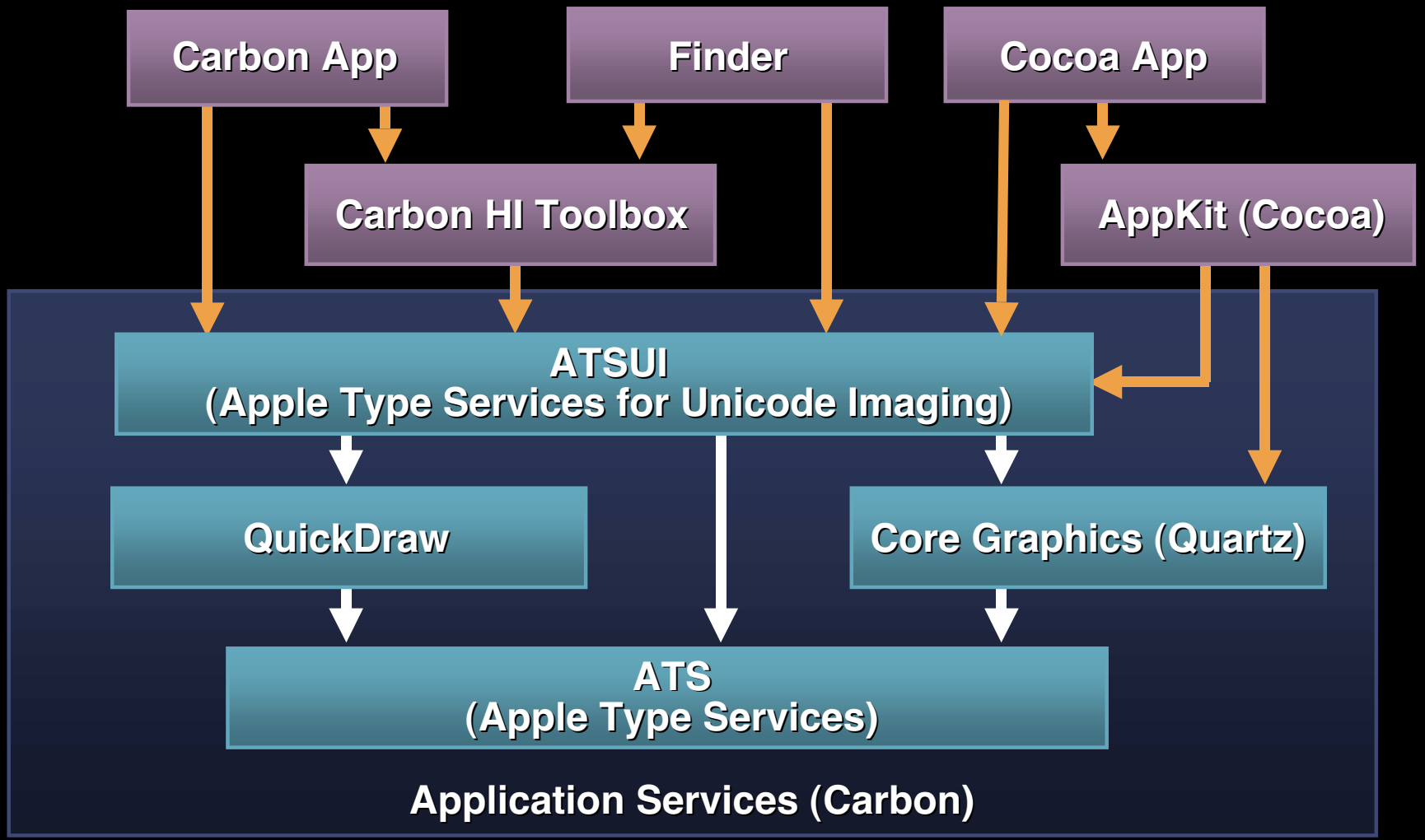
ATSUI and Quartz

- Fully integrated with Quartz
- Full Quartz anti-aliasing
- Respects settings in the CGContext
    - Color Space
    - Scaling
- Uses Quartz text rendering attributes

code
日本語


Zapfino
Beautiful Type

ATSUI Throughout Mac OS X



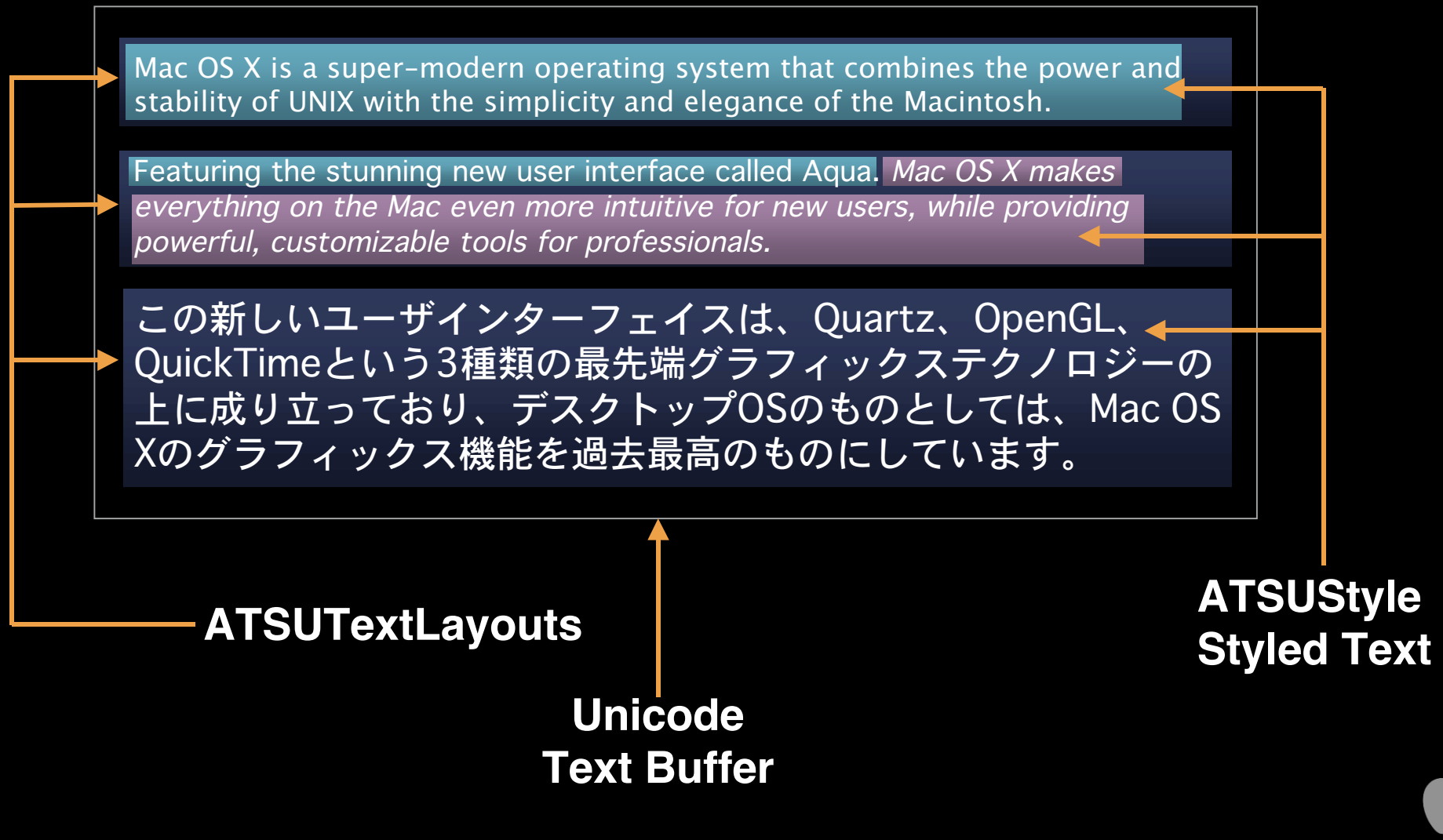
When Should I Use ATSUI Directly?

- Fine control
  - Glyph positions, text at an angle or on a path, line positioning
- Text features
  - Vertical, justification control, optical alignment, . . .
- Writing your own text editing engine
  - Can not use MLTE but need Unicode or ATSUI features

ATSUI Concepts

- ATSUStyle Objects
- ATSUTextLayout Objects

How Do the Objects Fit In?



ATSUStyle Objects

- Opaque object that represents a collection of stylistic attributes
- Never tied to a specific layout or run of text

What Can Be Stored in ATSUStyle?

- Font, size, color, vertical, with- and cross-stream shift, kern control, optical, hanging, . . .
- Font features (defined by font)
    - Ligatures, swashes, variant glyphs, . . .
- Font variations
    - Continuously variable weight, width, . . .

ATSUTextLayout Objects

- Ties Unicode text buffer with runs of ATSUStyle
- Keeps track of soft line breaks and tab stops
- Caches information about the text
- Customized through the use of Layout and Line controls

Layout and Line Controls

- Allow the control of line width, rotation, justification, flush, baseline, . . .
- Can be applied to an entire ATSUTextLayout or an individual line

Demo

New ATSUI Features

Tom Madden
ATSUI Tamer

New Features

- Batch line breaking
- Tab support
- Direct access
- Style flattening
- Variant glyphs
- Font Fallbacks Objects
- Thread safety
- Text measurement

Batch Line Breaking

- Paragraph based, like ATSUI
- Can reduce line breaking time by up to 50%
- Less code than ATSUBreakLine

Old: ATSUBreakLine Loop

```
ItemCount numSoftBreaks;
UniCharArrayOffset currentStart = 0;
UniCharArrayOffset currentEnd = textLength;
do {
        ATSUBreakLine(
          layout, currentStart, lineBreakWidth,
          true, &currentEnd);
        currentStart = currentEnd;
} while (currentEnd < textLength);
ATSUGetSoftLineBreaks(layout,
  kATSUFromTextBeginning, kATSUToTextEnd, 0,
  NULL, &numSoftBreaks);
```

New: ATSUBatchBreakLines

```
ATSUBatchBreakLines(layout,
  kATSUFromTextBeginning, kATSUToTextEnd,
  lineBreakWidth, &numSoftBreaks);
```

- Can be used only if the maximum line break width for all lines is the same

Tab Support

- Tab stops can now be set in an ATSUTextLayout object
    - ATSUSetTabArray
    - ATSUGetTabArray
- Simplifies the task of implementing rulers

Tab Support—New Data Types

- ATSUTab

```
struct ATSUTab {
  ATSUTextMeasurement tabPosition;
  ATSUTabType                 tabType;
};
```

- ATSUTabType
  - Left, right, and center tabs

ATSUI DirectAccess

- What is it?
- What can it be used for?
- How do I use it?

DirectAccess: What Is It?

- New set of API
    - Direct access to glyph data during the layout process
- New Callback Definition
    - Control over ATSUI's internal layout process

DirectAccess: What Can It Be Used For?

- Fine control over layout metrics
- Override ATSUI's internal layout operations
- Glyph replacement

Using DirectAccess

- Install the operation override callback for the operations you wish to override
    - ATSUSetLayoutControls with the kATSULayoutOperationOverrideTag tag
- Call ATSUI normally to measure and draw
- Callback invoked when re-layout is needed

ATSUI's Layout Operations

- Linguistic
    - Bi-directional level calculation
    - Glyph Morphing
        - Ligatures, contextual forms, etc.
- Kerning
- Tracking
- Baselines
- Justification

Which Operations Can I Override?

- Justification
- Morph
- Kerning
- Baseline
- Tracking
- Post Layout
    - To perform adjustment after ATSUI has finished all operations

Callback Implementation

```
OSStatus(*ATSUDirectLayoutOperationOverrideProcPtr)(
 ATSULayoutOperationSelector iCurrentOperation,
   ATSULineRef iLineRef,
   UInt32 iRefCon,
   void *iOperationCallbackParameterPtr,
   ATSULayoutOperationCallbackStatus *oCallbackStatus);
```

- Determine reason using iCurrentOperation
- Use iLineRef to get glyph data
- Tweak the glyph data
- Return call status in oCallbackStatus

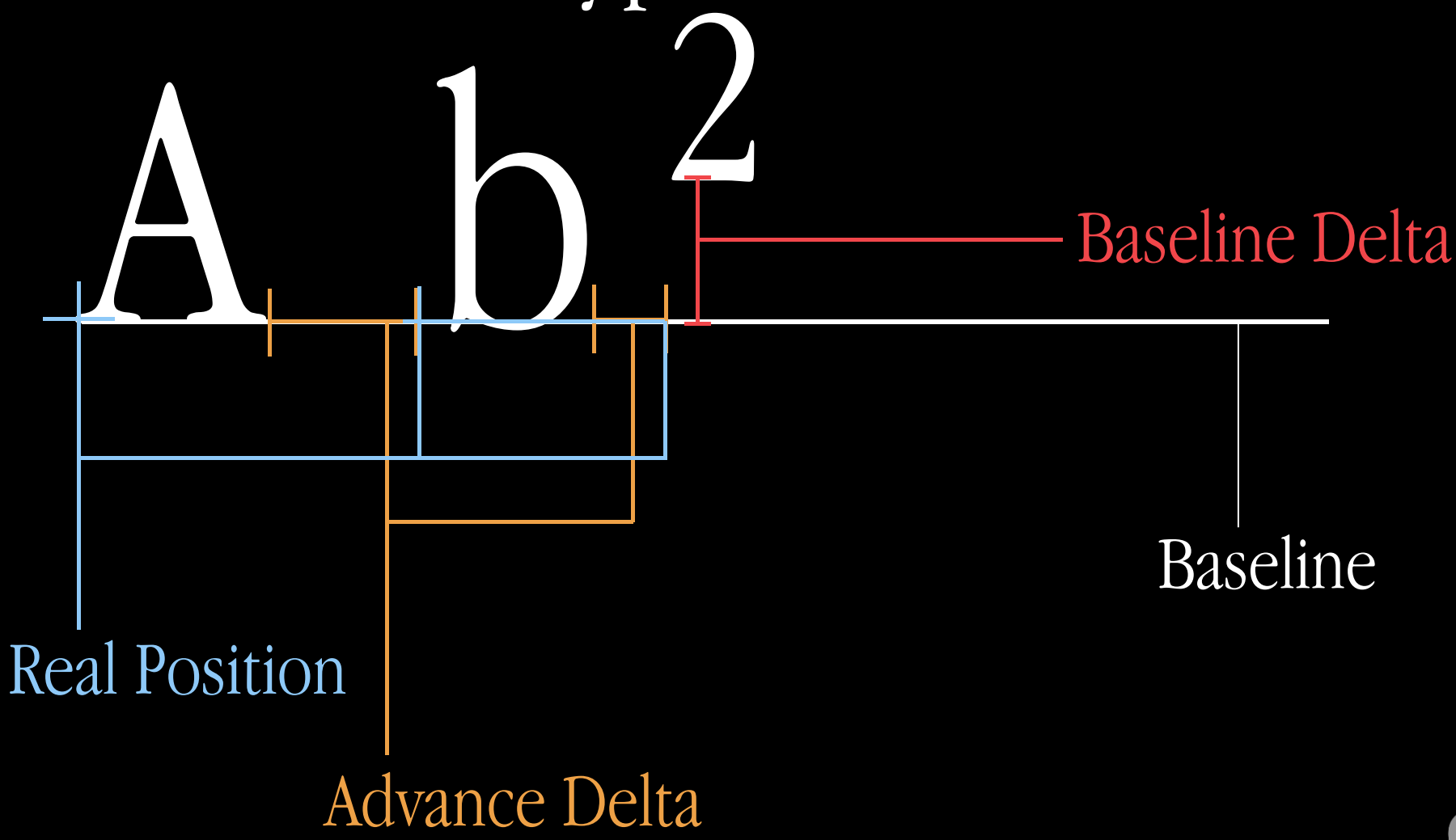
How Do I Get the Glyph Data From Inside the Callback?

- ATSUDirectGetLayoutDataArrayPtrFromLineRef
- Returns a direct pointer
- Fast
- Dispose of the data pointer by calling ATSUDirectReleaseLayoutDataArrayPtr

Can I Get Glyph Data Outside of the Callback?

- ATSUDirectGetLayoutDataArrayPtrFromTextLayout
- Returns a copy of the data
- Dispose of the data pointer by calling ATSUDirectReleaseLayoutDataArrayPtr

What Data Makes an ATSUI Glyph?



What Data Makes an ATSUI Glyph?

- ATSLayoutRecord

```
struct ATSLayoutRecord {
        ATSGlyphRef             glyphID;
        ATSGlyphInfoFlags       flags;
        ByteCount               originalOffset;
        Fixed                   realPos;
        };
```

- Baseline Delta (Fixed)
- Advance Delta (Fixed)
- Style Index (UInt16)
- Device Delta (SInt16)

Demo

DirectAccess

Style Flattening

- Flatten style run data to a stream
- Reconstruct styles from a stream
- Standard Pasteboard format
    - ‘ustl’ version 2.0
    - Used by MLTE
    - Understood by Cocoa Applications
    - All structures defined publicly

Flattening Styles

- ATSUFlattenStyleRunsToStream
- Used to flatten multiple style runs into a stream of data
- Pass in an array of ATSUStyle objects, an array of run lengths for the associated ATSUStyle objects, and an allocated buffer

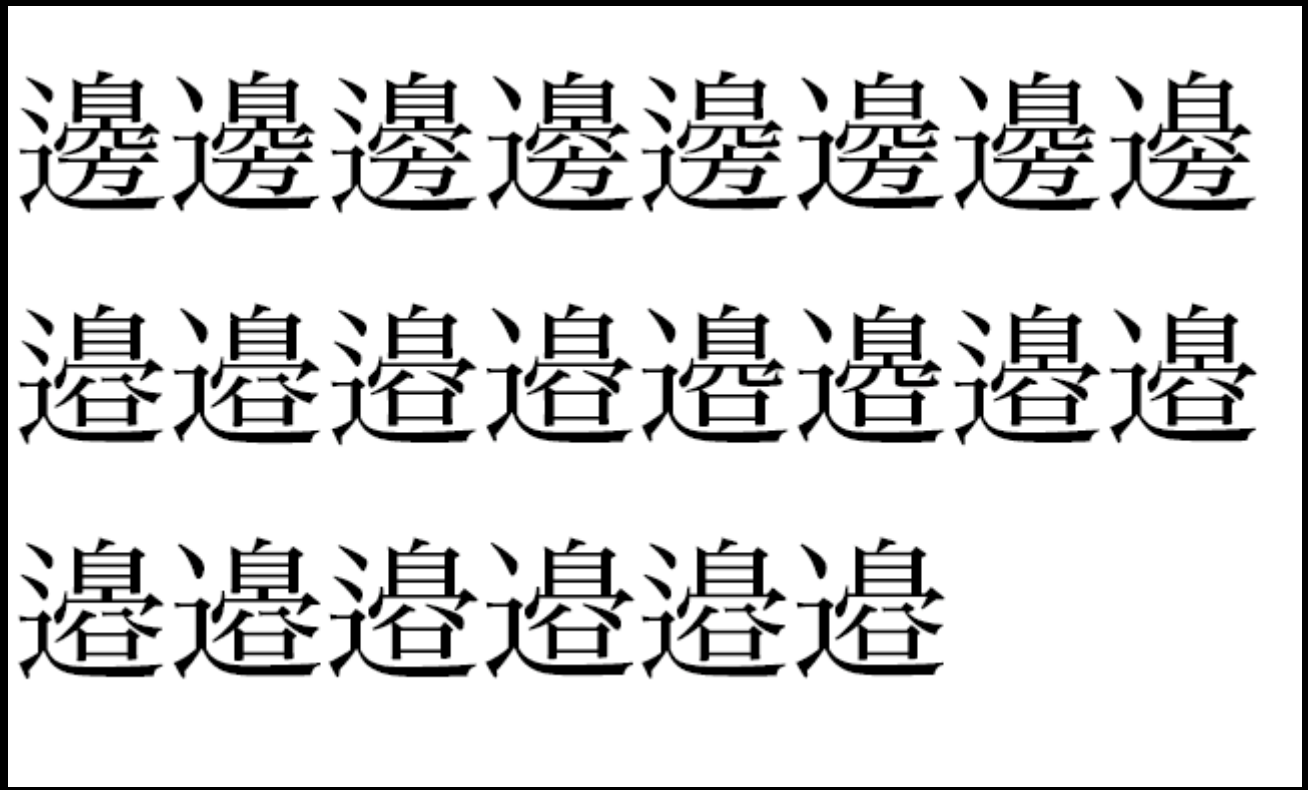
Reconstructing Style Runs

- ATSUUnflattenStyleRunsFromStream
- Returns the style objects and run information in two separate arrays
- Caller is responsible for disposing of the ATSUStyle objects created by this call
- Caller is responsible for matching up the style runs to the associated Unicode text
    - Usually exported to the pasteboard as ‘utxt’ data

Variant Glyphs

- Display glyphs that do not have an explicit Unicode character to glyph mapping
    - Variations of a glyph that does have a mapping
    - Access to characters in the fonts that otherwise would not be accessible

Variant Glyphs



Variant Glyph Support

- New ATSUStyle tag
  - kATSUGlyphSelectorTag
- New Attribute Structure
  - ATSUGlyphSelector
- Choose variant glyph by font specific glyph or CID
- Get variant glyph information from input method via TSM
  - kEventParamTextInputGlyphInfoArray text input event parameter

Font Fallback Objects

- Specifies a search order for font substitution
  - Reference to a list of ATSUFontIDs
- Attached to an ATSUTextLayout object
  - kATSULineFontFallbacksTag
- Safer than relying on the global font fallback list

Font Fallback Objects

- New API to create, destroy, and manage font fallback objects
    - ATSUCreateFontFallbacks
    - ATSUDisposeFontFallbacks
    - ATSUSetObjFontFallbacks
    - ATSUGetObjFontFallbacks
- Font fallback objects should be shared if possible

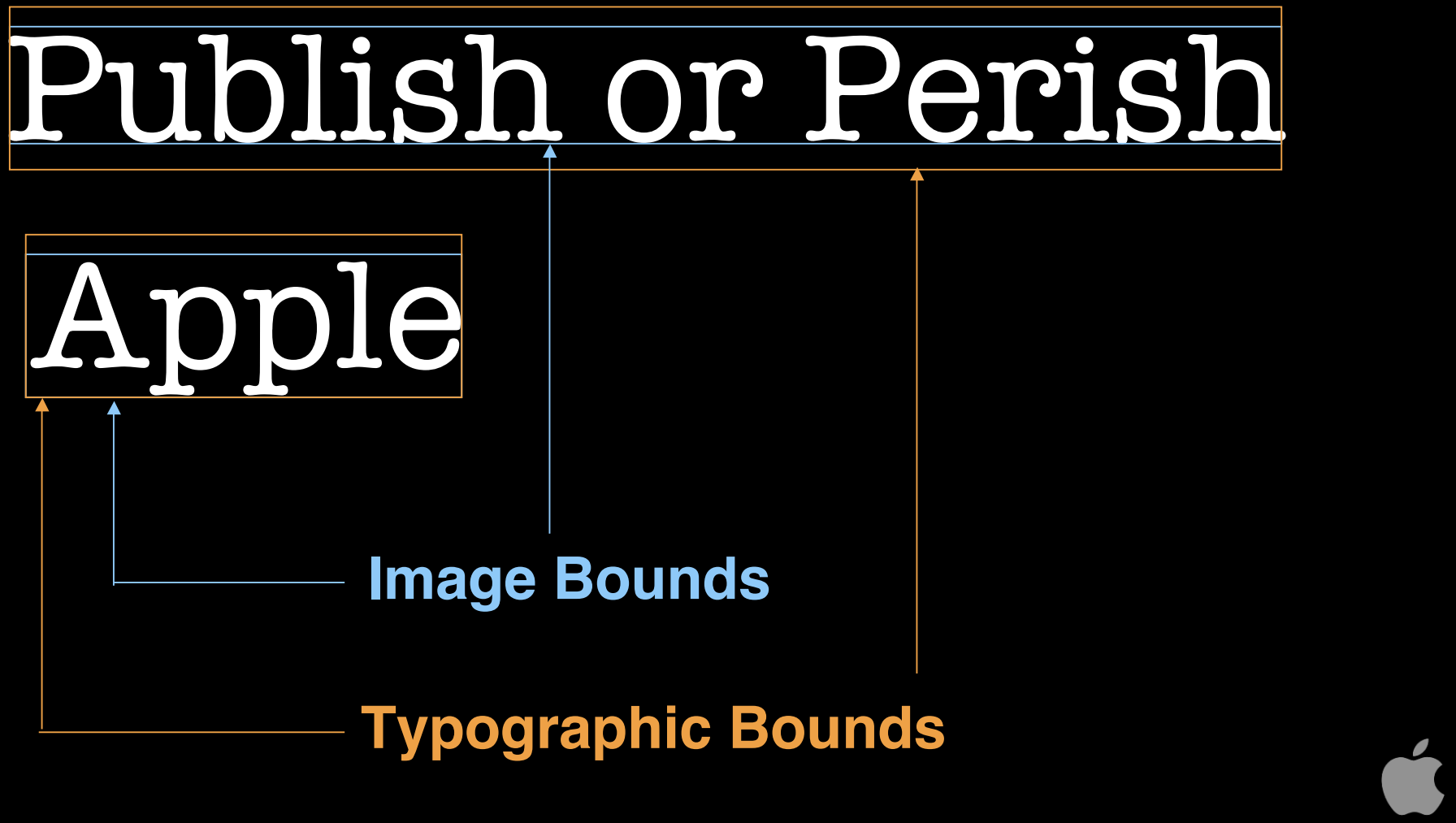
ATSUI's Thread Safety

- ATSUI objects are thread safe
- ATSUI APIs are thread safe
    - Be careful with global font fallback lists set by ATSUSetFontFallbacks

Text Measurement

- Review
    - Typographic vs. Image Bounds
    - Three APIs to measure
- New options

Text Measurement: Typographic vs. Image Bounds



Text Measurement: ATSUGetGlyphBounds

- Typographic bounds of the final laid out line
  - Returns bounding trapezoids
  - Aligned on fractional pixel boundaries
- Useful when laying out line after line of text
  - Prevents collisions with ascenders and descenders

Text Measurement: ATSUMeasureTextImage

- Image bounds of the final laid out line
  - Returns bounding rectangle
  - Aligned on integer, whole pixel boundaries
- Useful for figuring out the exact area in which ATSUI will draw

Text Measurement: ATSUMeasureText

- Typographic bounds of a line prior to final layout
  - Returns bounding rectangle based on ascent, descent, start point, and end point
  - Aligned on fractional pixel boundaries
  - Ignores any previously set line attributes such as rotation, flushness, justification, etc.
- Will likely force an extra layout operation
- Useful for calculating your own line breaking

Text Measurement: New Ascent and Descent Options

- Ascent Tag: kATSULineAscentTag
  - Gets or sets the maximum typographical ascent of all fonts used on a line or layout
- Descent + Leading Tag: kATSULineDescentTag
  - Gets or sets the maximum typographical descent + leading of all fonts used on a line or layout
- Use with ATSUGetLineControl or ATSUGetLayoutControl
- The most efficient way to get the ascent or descent + leading on Jaguar

ATSUI
Dos and Don’ts

Aaron Haney

Layouts

- **Don't** create a layout for each word, style run, or line
- **Do** create a layout for each paragraph

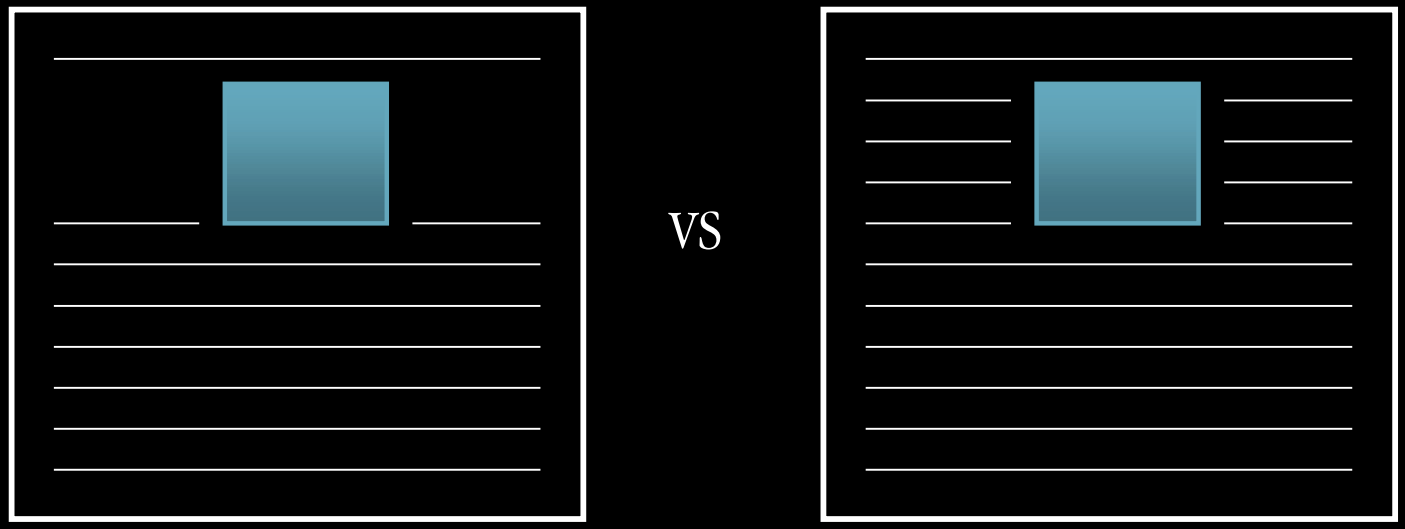
Layouts (Cont.)

- **Don't** throw away text layouts when text is altered
- **Do** use ATSUSetTextPointerLocation, ATSUTextDeleted, and ATSUTextInserted

Layouts (Cont.)

- **Don't** use multiple layouts to flow text around complex shapes
- **Do** use multiple lines and vary the line width

Example of flowing text around complex shapes:



Line Breaking

- **Don't** loop over ATSUBreakLine unless it is absolutely necessary
- **Do** use the new ATSUBatchBreakLines call

Line Breaking (Cont.)

- **Don't** set soft line breaks manually unless it is absolutely necessary
- **Do** pass true as the iUseAsSoftLineBreak parameter to ATSUBreakLine whenever possible

Styles

- **Don't** recreate style objects for each draw
- **Do** keep style objects around as long as they might be needed

CG Drawing

- **Don't** create and destroy the CGContext at each draw unless it is absolutely necessary
- **Do** retain the same CGContext until all drawing is completed

CG Drawing (Cont.)

- **Don't** use CreateCGContextForPort to do CG drawing
- **Do** use QDBeginCGContext and QDEndCGContext

Measuring

- **Don't** use ATSUMeasureText to get the ascent and descent of a line
- **Do** use ATSUGetGlyphBounds; Or, on Jaguar or later, use kATSULineAscentTag and kATSULineDescentTag

Font Fallbacks

- **Don't** use global font fallbacks
- **Do** use the per-layout font fallback objects (ATSUFontFallbacks)

Font Fallbacks (Cont.)

- **Don't** throw away ATSUFontFallbacks objects
- **Do** keep them around as long as possible, and even share them between layouts

Object Sharing

- **Don't** share ATSUTextLayout objects between threads, unless running under Jaguar
- **Do** share ATSUStyle and ATSUFontFallback objects between threads

Summary

Xavier Legros

Summary

- ATSUI brings developers:
  - Advanced Unicode text drawing
  - Advanced Typography
  - Support for new languages

Documentation

- Apple Type Services for Unicode Imaging Reference
  - **http://developer.apple.com/techpubs/macosx/Carbon/text/ATSUI/atsui.html**
  - Covers ATSUI 1.2
- Updated documentation for ATSUI 2.4
  - On web site in early summer
  - HTML and PDF

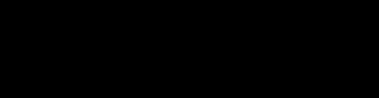
Roadmap

200 Making Your Application Unicode Savvy	Room C **Tue., 9:00am**
201 Font Manager	Room C **Tue., 10:30am**
208 MLTE: A Unicode Text Engine	Room A2 **Thurs., 9:00am**
010 Going International With Mac OS X	Room A2 **Fri., 10:30am**

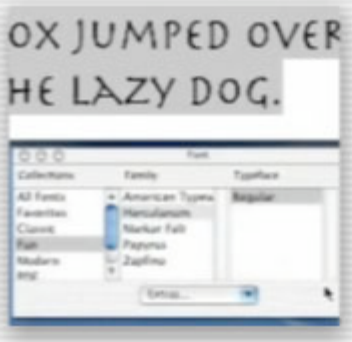
Who to Contact

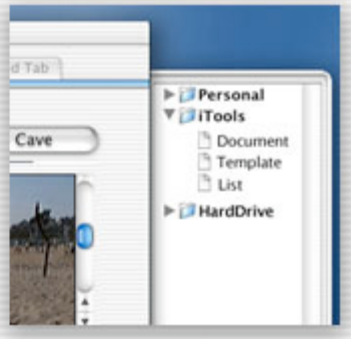
Xavier Legros
Mac OS X Evangelist
Apple Worldwide Developer Relations
xavier@apple.com

http://developer.apple.com/wwdc2002/urls.html



Q&A





Xavier Legros
Mac OS X Evangelist
xavier@apple.com

http://developer.apple.com/wwdc2002/urls.html


WWDC2002


WWDC2002


WWDC2002